Multitemporal conditional schema evolution

Jensen, Ole Guttorm; Bøhlen, Michael Hanspeter

Published in:
Conceptual Modeling for Advanced Application Domains

Publication date:
2004

Document Version
Publisher's PDF, also known as Version of record

Multitemporal Conditional Schema Evolution

Ole G. Jensen[1] and Michael H. Böhlen[2]

[1] Department of Computer Science, Aalborg University,
Fredrik Bajers Vej 7E, DK-9220 Aalborg Øst, Denmark,
guttorm@cs.aau.dk,
[2] Faculty pf Computer Science, Free University of Bozen-Bolzano,
Dominikanerplatz 3, I-39100 Bolzano, Italy,
boehlen@inf.unibz.it

Abstract. Schema evolution is the ability of the database to respond to changes in the real world by allowing the schema to evolve. The *multidimensional conditionally evolving schema*(MD-CES) is a conceptual model for conditional schema changes, which modify the schema of those tuples that satisfy the change condition. The MD-CES is lossless and preserves schemas, but has an exponential space complexity. In this paper we restrict conditional schema changes to timestamp attributes. Specifically, we develop 1D-CES for schema versioning over one time dimension, and 2D-CES for schema versioning over two time dimensions. We show that the space complexity of these new evolution models is linear or polynomial. 1D-CES and 2D-CES are compared to temporal schema versioning, and we show that, unlike valid time versioning, they are lossless and achieve the same space complexity as temporal versioning if the schema changes are ordered.

1 Introduction

Conditional schema changes modify the schema of those tuples that satisfy the change condition [5], and they properly subsume unconditional and temporal schema changes. The semantics of conditional schema evolution is defined in terms of the *multi-dimensional conditionally evolving schema* (MD-CES) [4]. The MD-CES has many desirable properties that are essential for relations that contain tuples with different intended schemas. First, the MD-CES is *lossless*, i.e., the intended schema of each tuple is consistent with the initial schema definition and the subsequent conditional schema changes. Thus, at any point during the evolution process the correct intended schema of a tuple is known.[3] Second, the MD-CES is *schema preserving*, i.e., all schemas are preserved and tuples never have to be migrated to a schema with more or less attributes. On the downside, however, the space complexity of the MD-CES grows exponentially with the number of schema changes, and, consequently, the MD-CES is not a practical model.

In this paper, we specialize conditional schema changes to conditions over timestamp attributes. This leads to the 1D-CES and 2D-CES with linear or polynomial space

[3] Models with an ad-hoc approach to schema evolution often violate this property. For example if NULL values are used to "flag" attributes that are not part of the schema and with multiple schema changes the correct identification of current and past intended schemas is often a problem.

complexity for monoconditional and biconditional schema changes, respectively. We prove that 1D-CES and 2D-CES are still lossless and, if the history of conditional schema changes is ordered, ensure schema preservation. We also provide the cost of ensuring schema preservation for 1D-CES and 2D-CES with unordered histories. Thus, by restricting the MD-CES to mono- and biconditional schema changes, we get a practical model for schema evolution that is particularly relevant to temporal schema versioning.

An important aspect of our work is that it offers an alternative approach to investigate the semantics of temporal evolution models. As pointed out before the semantics of temporal schema versioning, in particular for bitemporal databases, is non-trivial. We compare 1D-CES and 2D-CES to transaction-time, valid-time, and bitemporal schema versioning. For transaction-time the two models are equivalent. In valid-time and bitemporal schema versioning, a schema change is applied to a single schema version specified by the user irrespective of the validity of the change and the version. In contrast, with MD-CES a schema change modifies the schema of all segments with a validity that overlaps the validity of the change. We show that valid-time and bitemporal schema versioning are not lossless, and that valid-time schema versioning does not ensure schema preservation.

2 Preliminaries

2.1 Multi-Dimensional Conditionally Evolving Schemas

A *multi-dimensional conditionally evolving schema* (MD-CES), $E = \{S_1, \ldots, S_n\}$, generalizes a relation schema and is defined as a set of schema segments. A *schema segment*, $S = (\mathcal{A}, P)$, consists of a schema $\mathcal{A}$ and a qualifier P. Throughout, we write $\mathcal{A}_S$ and P_S to directly refer to the schema and qualifier of segment S, respectively. As usual, a *schema*, $\mathcal{A} = \{A_1, \ldots, A_n\}$, is defined as a set of attributes. No distinction is made between schemas and sets of attributes. A *qualifier* P is either TRUE, FALSE, or a conjunction/disjunction of attribute constraints. An *attribute constraint* is a predicate of the form $A\theta c$ or $\neg(A\theta c)$, where A is an attribute, $\theta \in \{<, \leq, =, \neq, \geq, >\}$ is a comparison predicate, and c is a constant. A MD-CES may have segments with different schemas. Consequently, some tuples may be missing attributes that appear in other segments. In order to evaluate attribute constraints on such tuples, $A\theta c$ is an abbreviation for $\exists v(A/v \in t \wedge v\theta c)$ where t is a tuple and A/v is an attribute/value pair. Likewise, $\neg(A\theta c)$ is an abbreviation for $\neg\exists(A/v \in t \wedge v\theta c)$. Note that this implies that the constraints $\neg(A = c)$ and $A \neq c$ are not equivalent.

A tuple t is a set of attribute values where each attribute value is an attribute/value pair: $\{A_1/v_1, \ldots, A_n/v_n\}$. The value must be an element of the domain of the attribute, i.e., if $dom(A)$ denotes the domain of attribute A, then $\forall A, v, t(A/v \in t \Rightarrow v \in dom(A))$. A tuple t *qualifies* for a segment S, $qual(t, S)$, iff t satisfies the qualifier P_S. A tuple satisfies a qualifier, $P(t)$, iff the qualifier is TRUE or the tuple makes the qualifier true under the standard interpretation. If a tuple t qualifies for a segment S in a MD-CES E, then $\mathcal{A}_S$ is the *intended schema* of t, i.e., $\forall t, S, E(S \in E \wedge qual(t, S)) \Rightarrow is(t, E) = \mathcal{A}_S)$. A tuple t *matches* a segment S iff the schema of S and t are identical: $match(t, S)$ iff $\forall A(A \in \mathcal{A}_S \Leftrightarrow \exists v(A/v \in t))$. If a tuple t

matches a segment S in the MD-CES E, then $\mathcal{A}_S$ is the *recorded schema* of t, i.e., $\forall t, S, E(S \in E \wedge match(t, S) \Rightarrow rs(t, E) = \mathcal{A}_S)$.

2.2 Conditional Schema Changes

A *conditional schema change* is an operation that changes the set of segments of a MD-CES. The condition determines the tuples that are affected by the schema change. A *condition* C is either TRUE, FALSE, an attribute constraint, or a conjunction of attribute constraints. For the purpose of this section we consider two conditional schema changes: adding an attribute, $\alpha(A, E, C)$, and deleting an attribute, $\beta(A, E, C)$. An extended set of schema changes that includes mappings between attributes and a discussion of their completeness can be found elsewhere [4].

$\alpha(A, E, C)$: An attribute A is added to the schemas of all segments that do not already include the attribute. For each such segment two new segments are generated: a segment with a schema that does not include the new attribute and a segment with a schema that includes the new attribute. Segments with a schema that already includes A are not changed.

$\beta(A, E, C)$: The attribute A is deleted from the schemas of all segments that include the attribute. For each such segment two new segments are generated: a segment with a schema that still includes the attribute and a segment with a schema that does not include the attribute. Segments with a schema that does not include A are not changed.

The precise formal definitions of conditional attribute additions and deletions are given in Figure 1. Note that conditional schema changes properly subsume regular (i.e., unconditional) schema changes. It is possible to have the condition (TRUE) select the entire extent of a relation.

$$\alpha(A, \emptyset, C) = \emptyset$$
$$\alpha(A, \{(\mathcal{A}, P)\} \cup E, C) = \begin{cases} \{(\mathcal{A}, P)\} \cup \alpha(A, E, C) & \text{iff } A \in \mathcal{A} \\ \{(\mathcal{A} \cup \{A\}, P \wedge C), (\mathcal{A}, P \wedge \neg C)\} & \\ \cup \alpha(A, E, C) & \text{iff } A \notin \mathcal{A} \end{cases}$$
$$\beta(A, \emptyset, C) = \emptyset$$
$$\beta(A, \{(\mathcal{A}, P)\} \cup E, C) = \begin{cases} \{(\mathcal{A}, P)\} \cup \beta(A, E, C) & \text{iff } A \notin \mathcal{A} \\ \{(\mathcal{A} \setminus \{A\}, P \wedge C), (\mathcal{A}, P \wedge \neg C)\} & \\ \cup \beta(A, E, C) & \text{iff } A \in \mathcal{A} \end{cases}$$

Fig. 1. Adding ($\alpha(A, E, C)$) and Deleting ($\beta(A, E, C)$) Attribute A on Condition C

A *history* $H = [\gamma_1(A_1, E, C_1), \ldots, \gamma_n(A_n, E, C_n)]$ where $\gamma_i \in \{\alpha, \beta\}$ is a sequence of conditional schema changes. We say that (E_0, H) is the *evolution history* of E where $E_0 = \{S\}$ consists of a single segment, iff $E = \gamma_n(A_n, \ldots \gamma_1(A_1, E_0, C_1) \ldots, C_n)$.

2.3 Lossless and Schema Preserving Properties

The MD-CES is both *lossless* [6] and *schema preserving* [5].

The lossless property ensures that the intended schema defined by a MD-CES for each tuple is consistent with the initial schema definition and the conditional schema changes. Intuitively, the intended schema of a tuple t can be determined from the initial schema by applying only those schema changes with a condition satisfied by t in the sequence given by the history.

The schema preserving property guarantees that all schemas defined by segments in a MD-CES also appear in segments after a conditional schema change. Schema preservation is a requirement if the segments of the MD-CES are to be used as relations to store the (heterogeneous) tuples of evolving relations.

3 Schema Change Conditions on a Single Attribute

Conditional schema evolution allows schema changes to be conditioned by any attribute in the schema. Since each conditional schema change potentially splits every segment in the MD-CES into two new segments, conditional schema evolution leads to MD-CESs where the number of segments is exponential in the size of the history [5]. Therefore, solutions that rely on the segments of a MD-CES are not tractable, including using the segments as relations to record tuples or querying individual segments.

In this section we restrict the conditions of conditional schema changes to a single attribute. This leads to *one-dimensional conditionally evolving schemas* (1D-CES). We show that the number of segments in a 1D-CES becomes proportional to the size of the history. This is achieved by the elimination of segments with false qualifiers. We show that this optimization preserves the lossless property of the MD-CES but can violate schema preservation. To achieve schema preservation additional segments must be kept in the 1D-CES increasing its space complexity to be polynomial in the size of the history. We show that if the history is ordered, then schema preservation is achieved without recording additional segments. First, however, we present a running example to be used throughout.

3.1 Running Example

We use a relation storing information about employees as a running example. The employee relation records a name, an address, and a phone number for each employee. Additionally, a timestamp attribute records when the fact was recorded in the database (a detailed discussion about different notions of time is deferred until Section 5). The schema $E_{\text{mployee}} = (N_{\text{ame}}, A_{\text{ddress}}, P_{\text{hone}}, T_{\text{ime}})$ defines the (initial) employee relation schema.

A conditional schema change $\gamma_1 = \alpha(S_{\text{n}}, E_{\text{mployee}}, T_{\text{ime}} \geq \text{2003-01-01})$ adds a social security number to employees recorded on and after January 1st 2003. The schema change splits the employee schema into two segments: $S_1 = (\{N, A, P, T\}, T < \text{20030101}\})$ and $S_2 = (\{N, A, P, S, T\}, T \geq \text{20030101}\})$.

The schema change only adds a social security number to the schema of employees recorded with a timestamp value equal to or greater than 2003-01-01. Employees recorded with a timestamp before 2003-01-01 are not affected.

We require that all tuples have a timestamp value and that schema changes cannot drop the timestamp attribute from the schema. The evaluation of qualifiers is affected by this requirement. Recall that qualifiers are existentially quantified, so $\neg(T\theta c) \equiv \neg\exists T(T\theta c)$. This implies that conditions $\neg(T = c)$ and $T \neq c$ are not equivalent. E.g. tuples without a T attribute value evalues to true for the former condition and false for the latter condition. However, since we require that all tuples *have* a timestamp value, the equivalence holds, and the negation can be pushed down to the comparison predicate as is the case for S_1.

A second schema change drops the address from the schema of employees recorded from the 1st of March 2003 and replaces it with a C_{ity} attribute. In response, two conditional schema changes are applied to the E_{mployee} MD-CES:

$$\beta(A_{\text{ddress}}, E_{\text{mployee}}, T_{\text{ime}} \geq \text{2003-03-01}) \qquad \alpha(C_{\text{ity}}, E_{\text{mployee}}, T_{\text{ime}} \geq \text{2003-03-01})$$

The conditional schema change on the left drops the A_{ddress} attribute from the schema of employees with a timestamp value of at least 2003-03-01. The conditional schema change on the right adds the C_{ity} attribute on the exact same condition. Both conditional schema changes are applied in sequence as seen below. Note that the order in which the two schema changes are applied does not change the result, because the schema changes affect different attributes (A_{ddress} and C_{ity} respectively).

$$E_2 = \alpha(C, \beta(A, E_1, T \geq \text{2003-03-01}), T \geq \text{2003-03-01}) =$$

$$\{(\{N, C, P, S, T\}, T \geq \text{2003-01-01} \wedge T \geq \text{2003-03-01} \wedge T \geq \text{2003-03-01})^1,$$
$$(\{N, P, S, T\}, T \geq \text{2003-01-01} \wedge T \geq \text{2003-03-01} \wedge T < \text{2003-03-01})^2,$$
$$(\{N, A, C, P, S, T\}, T \geq \text{2003-01-01} \wedge T < \text{2003-03-01} \wedge T \geq \text{2003-03-01})^3,$$
$$(\{N, A, P, S, T\}, T \geq \text{2003-01-01} \wedge T < \text{2003-03-01} \wedge T < \text{2003-03-01})^4,$$
$$(\{N, C, P, T\}, T < \text{2003-01-01} \wedge T \geq \text{2003-03-01} \wedge T \geq \text{2003-03-01})^5,$$
$$(\{N, P, T\}, T < \text{2003-01-01} \wedge T \geq \text{2003-03-01} \wedge T < \text{2003-03-01})^6,$$
$$(\{N, A, C, P, T\}, T < \text{2003-01-01} \wedge T < \text{2003-03-01} \wedge T \geq \text{2003-03-01})^7,$$
$$(\{N, A, P, T\}, T < \text{2003-01-01} \wedge T < \text{2003-03-01} \wedge T < \text{2003-03-01})^8\}$$

The employee MD-CES, E_2, now consists of eight segments (marked with small numbers in the above expression). Clearly, the number of segments in E_2 increases exponentially with the application of conditional schema changes.

3.2 Elimination of Segments with False Qualifiers

Consider the qualifier of segment 2 in E_2:

$$T \geq \text{2003-01-01} \wedge T \geq \text{2003-03-01} \wedge T < \text{2003-03-01} \Leftrightarrow$$
$$T \geq \text{2003-03-01} \wedge T < \text{2003-03-01} \Leftrightarrow$$
$$\text{FALSE}$$

Clearly, the qualifier is false. No tuple can qualify for such segments. Future conditional schema changes applied to a segment S with a false qualifier P_S cannot result in segments with non-false qualifiers, because those qualifiers are either identical to P_S or in conjunctive form with P_S (cf. Section 2) and, clearly, $\text{FALSE} \wedge p \equiv \text{FALSE}$.

A MD-CES defines the intended schema of all tuples for a given initial schema and history. The association of tuples and their intended schemas is done through qualification of tuples with individual segments. Since tuples cannot qualify for segments with false qualifiers, such segments define no intended schemas. We can therefore omit segments with false qualifiers from a MD-CES without loss to the set of intended schemas defined by it, i.e., without losing the lossless property. This is stated by Lemma 1. All proofs have been omitted due to space considerations, but can be found in [3].

Lemma 1. *Let E and E' be MD-CESs and let γ be a conditional schema change. If E' consists of exactly the non-false segments of E, then $\gamma(A, E, C)$ and $\gamma(A, E', C)$ define the exact same intended schema for any tuple, i.e., $\forall\gamma, t(is(t, \gamma(A, E, C)) = is(t, \gamma(A, E', C)))$.*

3.3 MD-CES with Linear Space Complexity

Schema changes conditioned by a single attribute result in a MD-CES, where the qualifier of each segment is a conjunction of predicates over that attribute only. We say that such MD-CESs are one-dimensional (denoted as 1D-CES). The intended schema of a tuple depends exclusively on its recorded attribute value for the attribute used to condition the schema changes. Moreover, for every value of that attribute, the 1D-CES defines exactly one intended schema, because the qualifiers of two segments part of the same MD-CES never overlap (a property of the MD-CES [5]).

Note that the number of segments in E_{mployee} scales with the number of conditional schema changes applied. In the general case conditional schema changes cause an exponential increase of segments, as each schema change potentially splits every segment into two new segments. However, because all the conditional schema changes applied to E_{mployee} are conditioned by the same attribute, at most one segment is split into two non-false segments after each schema change (every other segment yields only one non-false segment).

Unbounded Conditions We shall assume that all conditional schema changes have *unbounded* conditions, i.e. conditions of the form $A\theta c$ where $\theta \in \{\geq, >, <, \leq\}$. Bounded conditions can be specified easily. E.g. $\alpha(A, E, T \geq t_1 \wedge T \leq t_2)$ has a bounded condition affecting only tuples with a T value in the interval from t_1 to t_2. Schema changes with bounded conditions can split up to two different segments (one per predicate in the condition). However, any conditional schema change with a non-false bounded condition has an equivalent pair of unbounded conditions. Table 1 shows the equivalent conditional schema changes for attribute addition and deletion with bounded condition, respectively. Conditional schema changes applying to a single point, i.e. with a condition of the form $A = c$, is a specialization of a bounded condition (since $(A = c) \equiv (A \geq c \wedge A \leq c)$).

We can assume unbounded conditions without loss of generality. Unbounded conditions result in segments where the qualifier is satisfied for a contiguous set of attribute values (an interval). This facilitates the comparison between segments of a 1D-CES (and 2D-CES) and schema versions in Section 5. Next, we give the main result of this section.

Table 1. Conditional Schema Change Equivalences between Bounded and Unbounded Conditions

Bounded Condition	Unbounded Conditions
$\alpha(A, E, T \geq t_1 \wedge T \leq t_2)$	$\alpha(A, E, T \geq t_1)$, $\beta(A, E, T > t_2)$
$\beta(A, E, T \geq t_1 \wedge T \leq t_2)$	$\beta(A, E, T \geq t_1)$, $\alpha(A, E, T > t_2)$

Lemma 2. *Let both E and E' be a MD-CES. Let $H = [\gamma_1(A_1, E, C_1), \ldots, \gamma_n(A_n, E, C_n)]$ be a history where $\gamma_i \in \{\alpha, \beta\}$ and each condition C_i is unbounded and over the same attribute T. If $E' = \gamma_n(A_n, \ldots \gamma_1(A_1, E, C_1) \ldots, C_n)$ then E' has a number of additional non-false segments that is proportional to the size of the history, i.e. $|E'| - |E| \leq |H|$.*

Lemma 2 shows that solutions to conditional schema evolution based on the segments of the MD-CES become tractable when schema changes are conditioned by a single attribute.

3.4 Ordering of Histories

While Lemma 2 guarantees the scalability of the 1D-CES, a conditional schema change can potentially change the schema of every segment in the 1D-CES. The 1D-CES is no longer guaranteed to ensure schema preservation once segments with false qualifiers have been eliminated. Using the segments of the 1D-CES as relations to record the (heterogeneous) tuples of an evolving relation, requires that existing schemas are preserved for tuples already recorded in the database with those schemas. In the worst case, a conditional schema change results in a 1D-CES where all the schemas of the original segments appear in new segments with false qualifiers. Therefore, they would all need to be kept in addition to the new segments created by the schema change. An upper bound of $\frac{n^2+n}{2}$ segments, where n is the size of the history, would have to be kept to ensure schema preservation. In this section we show that by ordering the history it is possible to keep the number of segments linear.

Example 1. Consider a 1D-CES with segments: $(\{A, C, T\}, T \geq 3)$ and $(\{B, C, T\}, T < 3)$. Assume a conditional schema change that drops the C attribute on the condition $T \geq 1$. The result is a 1D-CES with the following segments:

1. $(\{A, T\}, T \geq 3 \wedge T \geq 1) \equiv (\{A, T\}, T \geq 3)$
2. $(\{A, C, T\}, T \geq 3 \wedge T < 1) \equiv (\{A, C, T\}, \text{FALSE})$
3. $(\{B, T\}, T < 3 \wedge T \geq 1)$
4. $(\{B, C, T\}, T < 3 \wedge T < 1) \equiv (\{B, C, T\}, T < 1)$

Note that the 1D-CES no longer contains a segment with the schema $\{A, C, T\}$ (since it was dropped due to a false qualifier).

Unbounded conditions over a single attribute leads to segments with qualifiers that are satisfied for a contiguous interval of attribute values. The problem illustrated by Example 1 arises when the condition of a schema change can be satisfied by attribute

values within the interval qualified by more than one segment. From Lemma 2 we have that at most one segment is split by the conditional schema change resulting in two non-false segments. For all other segments, the conditional schema change creates only one non-false segment. The interval of attribute values qualified by each of these segments either all satisfy the condition or none satisfy the condition. In the former case, the segment with the schema of the original segment, will have a false qualifier.

To ensure schema preservation while still eliminating segments with false qualifiers, we impose an ordering on the conditional schema changes. Definition 1 defines ordered histories of conditional schema changes.

Definition 1. (ordered history) *Let $H = [\gamma_1(A_1, E, C_1), \ldots, \gamma_n(A_n, E, C_n)]$ be a history. If each conditional schema change applies to a proper subset of the tuples which the previous conditional schema change applied to, i.e., if $\{t|t \in dom(T) \wedge C_{i+1}(t)\} \subset \{t|t \in dom(T) \wedge C_i(t)\}$ where $dom(T)$ is the domain of T attribute values, then H is an* ordered history.

In general, all schemas are preserved by a conditional schema change if it splits exactly one segment into two non-false segments and no attribute value in the interval defined by any other segment satisfy the condition. This will ensure that all previously defined schemas appear in segments with non-false qualifiers.

Lemma 3 states that a 1D-CES with an ordered history preserves all previously defined schemas.

Lemma 3. *Let $H = [\gamma_1(A_1, E, C_1), \ldots, \gamma_n(A_n, E, C_n)]$ be a history where $\gamma_i \in \{\alpha, \beta\}$ and each condition C_i is unbounded over a single attribute T. Let E be a MD-CES with a single segment, and let each $E_i = \gamma_i(A_i, \ldots \gamma_1(A_1, E, C_1) \ldots, C_i)$ be a MD-CES. If H is an ordered history then E_n preserves all schemas defined by segments in E and E_1 to E_{n-1}, i.e., H is ordered implies that $\forall S(S \in E \cup E_1 \cup \ldots \cup E_{n-1} \wedge \exists S'(S' \in E_n \wedge \mathcal{A}_S = \mathcal{A}_{S'}))$.*

Lemma 2 and 3 ensure that a 1D-CES with an ordered history of conditional schema changes using unbounded conditions over a single attribute is both lossless and schema preserving and has a space complexity, which is linear in the size of the history.

4 Schema Change Conditions on Different Attributes

So far conditional schema changes with a single attribute constraint have been considered. We refer to these schema changes as monoconditional schema changes. This section investigates histories where monoconditional schema changes are over different attributes. We show that as the number of different attributes used in the conditions increases, so does the space complexity of the MD-CES. We also consider biconditional schema changes, i.e., schema changes where the condition is a conjunction of two attribute constraints over different attributes. Biconditional schema changes lead to *two-dimensional conditionally evolving schemas* (2D-CES). A 2D-CES with a history of biconditional schema changes has a polynomial space complexity. We show that an ordered history of biconditional schema changes ensures a 2D-CES, which is lossless and schema preserving and has a linear space complexity.

Monoconditional schema changes over different attributes are orthogonal. These conditional schema changes split every segment in a MD-CES into two non-false segments. Since the qualifier P of the segment S and the condition C of the change are over different attributes, the logical conjunction $P \wedge C$ cannot be equivalent to false (unless either P or C is already false), so both segments resulting from applying the conditional schema change to S have non-false qualifiers.

Lemma 4 establishes the upper bound on the number of segments with non-false qualifiers in a MD-CES with a history of monoconditional schema changes over different attributes.

Lemma 4. *Let H be a history of monoconditional schema changes with unbounded conditions. Let $\mathcal{A}_H = \{A_1, \ldots, A_n\}$ be the set of attributes appearing in conditions in H, and let $num(A, H)$ be the number of schema changes in H conditioned by A. Let E and E' be a MD-CES and let E consist of a single segment. If (E, H) is the evolution history of E' then the upper bound on the number of segments with non-false qualifiers in E' is $(1 + num(A_1, H)) \times \ldots \times (1 + num(A_n, H))$.*

4.1 Ordering of Biconditional Schema Changes

In the general case, biconditional schema changes lead to a polynomial number of segments in a 2D-CES. This occurs when the condition of a schema change overlaps the qualifiers of all segments. However, if the condition is contained by the qualifier of a single segment then the number of segments increase by at most one.

This is the case for ordered histories of conditional schema changes. Recall that a history is ordered iff each conditional schema change applies to a proper subset of the tuples which the previous conditional schema change applied to. For biconditional schema changes this occurs when the conditional of each schema change is contained by the qualifier of the latest segment.

Lemma 5 states that the number of segments in a MD-CES defined by a history of biconditional schema changes is proportional to the size of the history, if there exists a sequence of those schema changes such that the sequence is an ordered history.

Lemma 5. *Let H be a history of biconditional schema changes with conditions C_i of the form $T \geq t_i \wedge V \geq v_i$. Let E and E' be a MD-CES and let E consist of a single segment. Let (E, H) be the evolution history of E'. If there exists a sequence H' of the schema changes in H such that H' is an ordered history, then E' has a number of additional non-false segments that is proportional to the size of the history, i.e., $|E'| - |E| \leq |H|$.*

Lemma 5 does not guarantee that all schemas are preserved by segments in the 2D-CES after a biconditional schema change has been applied. Schema preservation is achieved by ordered histories as stated in Lemma 6.

Lemma 6. *Let $H = [\gamma_1(A_1, E, C_1), \ldots, \gamma_n(A_n, E, C_n)]$ be a history of biconditional schema changes where $\gamma_i \in \{\alpha, \beta\}$. Let E be a MD-CES with a single segment, and let each $E_i = \gamma_i(A_i, \ldots \gamma_1(A_1, E, C_1) \ldots, C_i)$ be a MD-CES. If H is ordered then E_n preserves all schemas defined by segments in E and E_1 to E_{n-1}, i.e., H is ordered implies that $\forall S(S \in E \cup E_1 \cup \ldots \cup E_{n-1} \wedge \exists S'(S' \in E_n \wedge \mathcal{A}_S = \mathcal{A}_{S'}))$.*

5 Related Work

A versioning approach to schema evolution has been proposed within the context of both OODBs and temporal databases. In OODBs, a new version of the object instances is constructed along with a new version of the schema. The *Orion* [1] schema versioning mechanism keeps versions of the whole schema hierarchy instead of the individual classes or types. Every object instance of an old schema can be copied or converted to become an instance of the new schema. The class versioning approach CLOSQL [8] provides update/backdate functions for each attribute in a class to convert instances from the format in which the instance is recorded to the format required by the application. The *Encore* [11] system provides exception handlers for old types to deal with new attributes that are missing from the instances. This allows new applications to access undefined fields of legacy instances.

Schema changes relating to time have been investigated in the context of temporal schema versioning, where proposals have been made for the maintenance of schema versions along one [7, 9, 10] or more time dimensions [2]. Two time dimensions are usually considered: *transaction time*, which tells when facts are logically present and events occur in the database, and *valid time*, which tells when facts are true and events occur in the reality [12].

In schema versioning each version associates a schema with its time pertinence specified as an interval of time stamp values. The symbol "0" denotes the special values *initiation* in transaction time (i.e. the time when the system was started) and *beginning* in valid time (i.e. the minimum value of valid time). The symbol "∞" denotes the special values *until_changed* in transaction time (which is used to timestamp a still current fact) and *forever* in valid time (i.e. the maximum value of valid time).

In this section we use 1D-CES and 2D-CES with mono- and biconditional schema changes over timestamp attributes as a yard stick to investigate temporal schema versioning. There is a strong similarity between segments in a MD-CES and schema versions. This facilitates a comparison between the effects of schema changes in both frameworks. Only changes at the intensional (schema) level are considered. The interaction between intensional and extensional versioning is considered elsewhere [5].

5.1 Transaction-time Schema Versioning

In transaction-time schema versioning, one of the versions is the *current* schema version. Only the current version can be affected by a schema change. When this occurs, the current version is archieved and replaced by a new current version obtained by applying the schema change to the old schema. The implicit transaction-time pertinence of a schema change is always $[now; \infty]$, i.e., the schema change takes effect when recorded in the database and remains in effect until changed by another schema change [2].

Example 2. Consider the schema version with schema (N, A, P, T) and time pertinence $[0; \infty[$. Two schema changes are applied: 1) on the $2003-01-01$ an S attribute is added to the schema, and 2) on the $2003-03-01$ the A attribute is dropped.

The first schema change results in a new current schema version with schema (N, A, P, S, T) and the time pertinence of the schema change. The old schema version is archieved and its time pertinence is restricted to $2003-01-01$:

$V_1 : (N, A, P, T)$ $[0; 2003-01-01[$
$V_2 : (N, A, P, S, T)$ $[2003-01-01; \infty[$

The second schema change is then applied to the new current schema version V_2. V_1 is not considered. The result is a new current schema version V_3 and the restriction of the time pertinence of V_2:

$V_1 : (N, A, P, T)$ $[0; 2003-01-01[$
$V_2 : (N, A, P, S, T)$ $[2003-01-01; 2003-03-01[$
$V_3 : (N, P, S, T)$ $[2003-03-01; \infty[$

Note the similarity between the time pertinence of schema versions in Example 2 and the qualifiers of segments in 1D-CES. Assuming that T records transaction-time, then both schema versions and segments define the intended schema for an interval a timestamp values.

The implicit time pertinence of schema changes in transaction-time schema versioning corresponds to unbounded monoconditional schema changes of the form $T \geq c_{now}$, where c_{now} is the timestamp value of now when the schema change is recorded.

Due to the nature of transaction-time, c_{now} will increase with each additional schema change. This implies that the history of schema changes in transaction-time schema evolution is always ordered.

Lemma 2 and 3 ensure that all schemas are preserved (so schema changes do not affect archieved schema versions) and only one schema (the current schema version) is affected by a schema change splitting it into a version with the old schema and a restricted time pertinence (corresponding to a conjunction with the negated condition for the schema change) and the new current schema version. Although, transaction-time schema versioning considers only the current version when a schema change is applied, the resulting schema versions are still equivalent to the segments in 1D-CES due to transaction-time providing a natural ordering of the schema changes.

5.2 Valid-time Schema Versioning

In valid-time schema versioning, a schema change creates a new schema version by applying the changes to a specified schema version. The new version is assigned the validity of the schema change. Previous schema versions completely overlapped by the validity of the new schema segment are deleted and schema versions which are only partially overlapped have their validity restricted accordingly [2].

The sequence of schema changes as well as the versions picked in each evolution step determine which schema versions are created.

Example 3. Consider the schema version with schema (N, A, P, V) and validity $[0; \infty[$. Two schema changes are applied: 1) attribute S is added with validity $[c_2; \infty[$, and 2) attribute C is added with validity $[c_1; \infty[$, where $c_1 = 2003-01-01$ and $c_2 = 2003-03-01$.

The table below shows the possible outcomes of applying the two schema changes. The sequence is the order in which the two attribute additions are applied. The second column indicates the result of applying the second schema change to the initial schema version, and the third column contains the schema versions resulting from applying the second schema change to the new version created by the first schema change.

Sequence	1st version		2nd version	
α_S, α_C	(N, A, P, V)	$[0; c_1[$	(N, A, P, V)	$[0; c_1[$
	(N, A, P, C, V)	$[c_1; \infty[$	(N, A, P, S, C, V)	$[c_1; \infty[$
α_C, α_S	(N, A, P, V)	$[0; c_1[$	(N, A, P, V)	$[0; c_1[$
	(N, A, P, C, V)	$[c_1; c_2[$	(N, A, P, C, V)	$[c_1; c_2[$
	(N, A, P, S, V)	$[c_2; \infty[$	(N, A, P, C, S, V)	$[c_2; \infty[$

Consider the schema changes in Example 3 isolated. According to the lossless property, we should expect that the S and C attributes both appear in the schema of tuples valid after $2003{-}03{-}01$, and that none of them are in the schema prior to $2003{-}01{-}01$. Additionally, between $2003-01-01$ and $2003-03-01$, only the C attribute should appear in the schema. This is achieved by three segments in 1D-CES regardless of the order in which the schema changes are applied.

In valid-time schema versioning, this is achieved by first adding the C attribute and then apply the addition of S to the new schema version resulting from applying the attribute addition of C. Note that in this case, the schema changes are ordered, and in each evolution step the schema change is applied to the schema version with a validity that extends forever (∞). The situation corresponds exactly to transaction-time schema versioning.

Valid-time schema versioning and monoconditional schema evolution differ on two points, when the conditions for transaction-time schema versioning are not met. First, there is no relation between the schema version chosen as the target for the schema change and the validity of the schema change. [4] Second, the schema change does not apply to any of the schema versions with a validity that overlaps the validity of the schema change. Instead, these schema versions are either deleted (if their validity is completely overlapped by the validity of the schema change) or restricted accordingly.

Valid-time schema versioning is not schema preservering. While it is possible for the number of schema versions to shrink as a result of a schema change (if the validity of the change overlaps several existing schema versions), the upper bound on the number of schema versions in valid-time schema versioning is proportional to the number of schema changes applied. The space complexity of 1D-CES and valid-time schema versioning are therefore the same.

5.3 Bitemporal Schema Versioning

In bitemporal schema versioning, schema versions are maintained along both transaction-time and valid-time. A schema change creates a new version by applying the changes

[4] There is, however, one case, where the schema version and the validity of the schema change are related. If no validity is specified for the schema change, then the schema change assumes the validity of the schema version.

to a specified schema version which is current. The new current version is assigned the validity of the schema change, and previous schema versions completely overlapped by the change validity are archieved, rather than being deleted as happens in valid-time schema versioning.

Example 4. Consider the schema version with the schema (N, A, P, T, V), time pertinence $[0; \infty[_t$, and validity $[0; \infty[_v$. First, an S attribute is added on $2003{-}02{-}01$ with validity $[2003{-}03{-}01; \infty[$. We are left with two current schema versions.

$$\begin{array}{lll} V_1 : (N, A, P, T, V) & [0; \infty[_t & [0; 2003{-}03{-}01[_v \\ V_2 : (N, A, P, S, T, V) & [2003{-}02{-}01; \infty[_t & [2003{-}03{-}01; \infty[_v \end{array}$$

Next, a C attribute is added on $2003-04-01$ with validity $[2003-01-01; \infty[$. The schema change is applied to a current version. Since both V_1 and V_2 are current, we choose one of them (V_1).

$$\begin{array}{lll} V_1 : (N, A, P, T, V) & [0; \infty[_t & [0; 2003{-}01{-}01[_v \\ V_2 : (N, A, P, S, T, V) & [2003{-}02{-}01; 2003{-}04{-}01[_t & [2003{-}03{-}01; \infty[_v \\ V_3 : (N, A, P, C, T, V) & [2003{-}04{-}01; \infty[_t & [2003{-}01{-}01; \infty[_v \end{array}$$

The validity of V_2 is completely overlapped by the validity of the schema change, so V_2 has to be archieved by updating its time pertinence.

Note that for bitemporal schema versioning "holes" can appear in the bitemporal domain, where no schemas are defined. E.g., the intended schema for a tuple recorded before $2003{-}02{-}01$ and valid after $2003{-}03{-}01$ cannot be determined for bitemporal schema versioning. Therefore, bitemporal schema versioning is not lossless. However, it is schema preserving, because it archieves the versions with a validity that is completely overlapped by the validity of the schema change rather than deleting them as in valid-time schema versioning. Only one new version is created in response to a schema change, so the space complexity of bitemporal schema versioning is linear.

6 Summary

The MD-CES is a conceptual model for conditional schema changes with many desirable properties that are essential for relations that tuples with different intended schemas. The MD-CES is both lossless and schema preserving, but has an exponential space complexity.

The paper investigates MD-CES where conditional schema changes are restricted to conditions over one or two timestamp attributes. This leads to 1D-CES and 2D-CES with a linear or polynomial space complexity. Both 1D-CES and 2D-CES are lossless and if the conditional schema changes in their histories are ordered, then they are also schema preserving.

Figure 2 compares 1D-, 2D-, and MD-CSE with transaction-time and valid-time. Ordered and unordered refers to whether the history of conditional schema changes is ordered or not, and ordered reordering refers to unordered 2D-CES, where there exists a different ordered sequence of the same conditional schema changes.

Table 2. Properties of Different Evolving Schema Models

	Lossless	Schema Preserving	Space Complexity
Transaction-time Schema Versioning	*Yes*	*Yes*	$O(n)$
Ordered 1D-CES	*Yes*	*Yes*	$O(n)$
Valid-time Schema Versioning	*No*	*No*	$O(n)$
Unordered 1D-CES	*Yes*	*No*	$O(n)$
Unordered 1D-CES with schema preservation	*Yes*	*Yes*	$O(n^2)$
Bitemporal Schema Versioning	*No*	*Yes*	$O(n)$
Unordered 2D-CES	*Yes*	*No*	$O(n^2)$
Unordered 2D-CES with ordered reordering	*Yes*	*No*	$O(n)$
Ordered 2D-CES	*Yes*	*Yes*	$O(n)$
MD-CES	*Yes*	*Yes*	$O(2^n)$

References

1. J. Banerjee, W. Kim, H.-J. Kim, and H.F. Korth. Semantics and Implementation of Schema Evolution in Object-Oriented Databases. In *ACM SIGMOD International Conference on Management of Data*, pages 311–322. ACM Press, 1987.
2. C.D. Castro, F. Grandi, and R.R. Scalas. Schema Versioning for Multitemporal Relational Databases. *Information Systems*, 22(5):249–290, 1997.
3. O.G. Jensen. *Multi-Dimensional Conditional Schema Evolution in Relational Databases*. PhD thesis, Aalborg University, 2004.
4. O.G. Jensen and M.H. Böhlen. Evolving Relations. In *Database Schema Evolution and Meta-Modeling*, volume 9th International Workshop on Foundations of Models and Languages for Data and Objects of *Springer LNCS 2065*, page 115 ff., 2001.
5. O.G. Jensen and M.H. Böhlen. Current, Legacy, and Invalid Tuples in Conditionally Evolving Databases. In *ADVIS*, volume Second International Conference, ADVIS 2002, Izmir, Turkey, October 23-25, 2002, Proceedings of *Springer LNCS 2457*, pages 65–82, 2002.
6. O.G. Jensen and M.H. Böhlen. Lossless Conditional Schema Evolution. In *ER*, volume 22nd International Conference on Conceptual Modeling, ER 2004, Shanghai, China, November 8-12, 2004, Proceedings, 14 pages, 2004.
7. L.E. McKenzie and R.T. Snodgrass. Schema Evolution and the Relational Algebra. *Information Systems*, 15(2):207–232, 1990.
8. Simon R. Monk and Ian Sommerville. Schema Evolution in OODBs using Class Versioning. *SIGMOD Record*, 22(3):16–22, 1993.
9. J.F. Roddick. SQL/SE - A Query Language Extension for Databases Supporting Schema Evolution. *ACM SIGMOD Record*, 21(3):10–16, 1992.
10. J.F. Roddick and R.T. Snodgrass. *Schema Versioning. In: The TSQL92 Temporal Query Language*. Noewell-MA: Kluwer Academic Publishers, 1995.
11. Andrea H. Skarra and Stanley B. Zdonik. The Management of Changing Types in an Object-Oriented Database. In *OOPSLA, 1986, Portland, Oregon, Proceedings*, pages 483–495, 1986.
12. R.T. Snodgrass et al. TSQL2 Language Specification. *ACM SIGMOD Record*, 23(1), 1994.